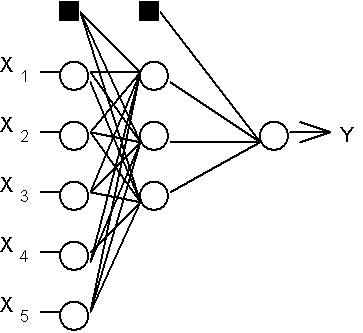
**Лабораторная работа №8**

## Метод обратного распространения ошибки

Структура многослойного персептрона с пятью входами, тремя нейронами в скрытом слое, и одним нейроном выходного слоя.



**Алгоритм обратного распространения ошибки.**

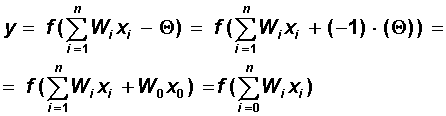
|  |  |
| --- | --- |
| Шаг 0. | Начальные значения весов всех нейронов всех слоев V(t=0) и W(t=0) полагаются случайными числами. |
| Шаг 1. | Сети предъявляется входной образ X, в результате формируется выходной образ y Y. При этом нейроны последовательно от слоя к слою функционируют по следующим формулам:  скрытый слой  http://alife.narod.ru/lectures/neural/Image29.gif  выходной слой  http://alife.narod.ru/lectures/neural/Image30.gif  Здесь f(x) - сигмоидальная функция, определяемая по формуле (6.1) |
| Шаг 2. | Функционал квадратичной ошибки сети для данного входного образа имеет вид:  http://alife.narod.ru/lectures/neural/Image31.gif  Данный функционал подлежит минимизации. Классический градиентный метод оптимизации состоит в итерационном уточнении аргумента согласно формуле:  http://alife.narod.ru/lectures/neural/Image32.gif  Функция ошибки в явном виде не содержит зависимости от веса Vjk, поэтому воспользуемся формулами неявного дифференцирования сложной функции:  http://alife.narod.ru/lectures/neural/Image33.gif  Здесь учтено полезное свойство сигмоидальной функции f(x): ее производная выражается только через само значение функции, f’(x)=f(1-f). Таким образом, все необходимые величины для подстройки весов выходного слоя V получены. |
| Шаг 3. | На этом шаге выполняется подстройка весов скрытого слоя. Градиентный метод по-прежнему дает:  http://alife.narod.ru/lectures/neural/Image34.gif  Вычисления производных выполняются по тем же формулам, за исключением некоторого усложнения формулы для ошибки  j.  http://alife.narod.ru/lectures/neural/Image35.gif  При вычислении  j здесь и был применен принцип обратного распространения ошибки: частные производные берутся только по переменным *последующего* слоя. По полученным формулам модифицируются веса нейронов скрытого слоя. Если в нейронной сети имеется несколько скрытых слоев, процедура обратного распространения применяется последовательно для каждого из них, начиная со слоя, предшествующего выходному, и далее до слоя, следующего за входным. При этом формулы сохраняют свой вид с заменой элементов выходного слоя на элементы соотвествующего скрытого слоя. |
| Шаг 4. | Шаги 1-3 повторяются для всех обучающих векторов. Обучение завершается по достижении малой полной ошибки или максимально допустимого числа итераций, как и в методе обучения Розенблатта. |

Как видно из описания шагов 2-3, обучение сводится к решению задачи оптимизации функционала ошибки градиентным методом. Вся “соль” обратного распространения ошибки состоит в том, что для ее оценки для нейронов скрытых слоев можно принять взвешенную сумму ошибок последующего слоя.

Параметр h имеет смысл темпа обучения и выбирается достаточно малым для сходимости метода. О сходимости необходимо сделать несколько дополнительных замечаний. Во-первых, практика показывает что сходимость метода обратного распространения весьма медленная. Невысокий тепм сходимости является “генетической болезнью” всех градиентных методов, так как локальное направление градиента отнюдь не совпадает с направлением к минимуму. Во-вторых, подстройка весов выполняется независимо для каждой пары образов обучающей выборки. При этом улучшение функционирования на некоторой заданной паре может, вообще говоря, приводить к ухудшению работы на предыдущих образах. В этом смысле, *нет* достоверных (кроме весьма обширной практики применения метода) гарантий сходимости.

Исследования показывают, что для представления произвольного функционального отображения, задаваемого обучающей выборкой, достаточно всего *два слоя* нейронов. Однако на практике, в случае сложных функций, использование более чем одного скрытого слоя может давать экономию полного числа нейронов.

В завершение лекции сделаем замечание относительно настройки порогов нейронов. Легко заметить, что порог нейрона может быть сделан эквивалентным дополнительному весу, соединенному с фиктивным входом, равным -1. Действительно, выбирая W0=, x0=-1 и начиная суммирование с нуля, можно рассматривать нейрон с нулевым порогом и одним дополнительным входом:



Дополнительные входы нейронов, соотвествующие порогам, изображены на Рис. темными квадратиками. С учетом этого замечания, все изложенные в алгоритме обратного распространения формулы суммирования по входам начинаются с нулевого индекса.

**Алгоритм:** ***BackPropagation*** (\eta, \alpha, \{x_i^d, t^d\}_{i=1,d=1}^{n,m}, NUMBER\_OF\_STEPS)

1. Инициализировать {*wij*}*i*,*j* маленькими случайными значениями, {Δ*wij*}*i*,*j* = 0
2. Повторить NUMBER\_OF\_STEPS раз:

Для всех d от 1 до m:

* 1. Подать \{x_i^d\}на вход сети и подсчитать выходы *oi* каждого узла.
  2. Для всех k \in Outputs

δ*k* = *ok*(1 − *ok*)(*tk* − *ok*).

* 1. Для каждого уровня l, начиная с предпоследнего:

Для каждого узла j уровня l вычислить

\delta _j = o_j(1 - o_j)\sum_{k \in Children(j)} \delta _k w_{j,k}.

* 1. Для каждого ребра сети {i, j}

Δ*wi*,*j* = αΔ*wi*,*j* + (1 − α)ηδ*joi*.

*wi*,*j* = *wi*,*j* + Δ*wi*,*j*.

1. Выдать значения *wij*.

где α - коэффициент инерциальнности для сглаживания резких скачков при перемещении по поверхности целевой функции